Test cases:

- Configs.da file is a list of configs of all test cases in a single file.
- It is sufficient run.da to execute all these test cases.
- Each test case will generate separate logger and ledger files categorized by test case number in the config file for all clients and Validators.

Structure of each test case:

```
{
'sq_num':#testcase(folders for ledgers and logs will be formed by this number),
'nclients': #Clients,
 'nreplicas': #Validators,
'nfaulty': #Fault,
'timeout': #Client timeout for retransmission,
'numMessages' : #number of transactions from each client,
'failure' : FailureConfig(failures = [

Failure(src='_',dest='leader',msg_type=MsgType.VOTE,round=1,prob=0.8,fail_type=FailType.M
sgLoss, val=9)],
seed = 12345678),

'delta': delta for validator local round timeouts (4*delta)
   }
```

Failure structure is maintained as explained in phase2.doc

Failure types:

- MsgLoss = Message Not Sent
- Delay = Delay in Sending a message
- SetAttr = highest_vote_round, current_round, high_qc.round
- ByzatineNoPropose = Byzantine Leader Doesn't Propose a Message

Note for all test cases logs location : logs/testcase.sq_num , ledgers location: ledgers/testcase.sq_num

Testcase3 (Happy Case):

```
 {'sq_num':3, 'nclients': 3, 'nreplicas': 7, 'nfaulty': 2, 'timeout': 5, 'numMessages' : 5, 'failure':
None, 'delta': 2},
```

Description: No Faults injected.
Expected Output: Ledger committing the given transaction and client getting acks.

TestCase 8 (Non Faulty Replicas Coming back to Sync amidst byzatine nodes):

```
{'sq_num':8, 'nclients': 5, 'nreplicas': 7, 'nfaulty': 2, 'timeout': 100, 'numMessages' : 2,
   'failure' : FailureConfig(failures = [
Failure(src=0,dest='leader',msg_type=MsgType.VOTE,round=3,prob=1,fail_type=FailType.MsgL
oss, val=15),
Failure(src=1,dest='leader',msg_type=MsgType.VOTE,round=3,prob=1,fail_type=FailType.MsgL
oss, val=15),
Failure(src=2,dest='leader',msg_type=MsgType.VOTE,round=3,prob=1,fail_type=FailType.MsgL
oss, val=15),
Failure(src=0,dest='leader',msg_type=MsgType.VOTE,round=4,prob=1,fail_type=FailType.MsgL
oss, val=15),
Failure(src=1,dest='leader',msg_type=MsgType.VOTE,round=4,prob=1,fail_type=FailType.MsgL
oss, val=15),
Failure(src=2,dest='leader',msg_type=MsgType.VOTE,round=4,prob=1,fail_type=FailType.MsgL
oss, val=15),
Failure(src=5,dest='_',msg_type='Propose',round=6,prob=1,fail_type=FailType.SetAttr,val=7,
   attr='highest_vote_round'),
Failure(src=6,dest='_',msg_type='Propose',round=6,prob=0.5,fail_type=FailType.SetAttr,val=7,
   attr='highest_vote_round')],seed = 12345678), 'delta': 2
   }
```

Description:
Fault Injection type: setting highest_vote_round to value that is greater than the current round, so it does not vote in that round.
In two consecutive rounds (in rounds 3 and 4) 3 out of 7 replicas will timeout, These are expected to form QC at the 6th round and sync with other replicas despite byzantine nodes 5 and 6 not participating in the 6th round due to Fault Injection.

Outcome:
3 and 4th rounds for the 3 replicas timeout and after getting consecutive QC's in 6th round because a quorum is formed with exactly 2f+1 (byzantine not participating) nodes the commit happens as expected.

Testcase 9 (Byzantine Leader):

```
{'sq_num':9, 'nclients': 5, 'nreplicas': 4, 'nfaulty': 1, 'timeout': 100, 'numMessages' : 2,
   'failure' : FailureConfig(failures = [
Failure(src=3,dest='',msg_type='',round=2,prob=1,fail_type=FailType.ByzatineNoPropose,val=N
one,
   attr=None)],seed = 12345678), 'delta': 2
   }
```

Description:
We injected a fault at round 2 where a byzantine leader will be elected and he does not propose any message after forming a QC.

Result: At that round, all validators proceed with a TC and get QC for that transaction in the next consecutive rounds

Testcase 10(Dedup for committed and retransmitted txns from the client):
```
{'sq_num':10, 'nclients': 5, 'nreplicas': 4, 'nfaulty': 1, 'timeout': 8, 'numMessages' : 2,
   'failure' : FailureConfig(failures = [
Failure(src=3,dest='',msg_type='',round=3,prob=1,fail_type=FailType.ByzatineNoPropose,val=N
one,
   attr=None)],seed = 12345678), 'delta': 2
   }
```

Description: We reduced the client timeout interval so that it timeout and retransmits the message as it did not receive the f+1 weak certificate

Result: The validator upon receive does not add that transaction to the mempool instead it checks in LRU committed transaction and ledger's speculated state tree to log that it had already committed/processing that particular transaction.

Note: we are not sending this message back to the client, we are just logging it in the log file like below and not doing any action for that transaction
```
2021-10-18 04:33:22 INFO     Already Received ['transactions_C4_0', 'C4', 4, 'C4_0'] at
Validator0
```

Testcase 11 (All Byzantine Nodes does not send any kind of message):

```
{'sq_num':11, 'nclients': 5, 'nreplicas': 4, 'nfaulty': 1, 'timeout': 8, 'numMessages' : 2,
   'failure' : FailureConfig(failures = [
Failure(src=3,dest='',msg_type='',round='_',prob=1,fail_type=FailType.MsgLoss,val=None,
   attr=None)],seed = 12345678), 'delta': 2
   }
```

Description: Since leader election is round-robin 1 in n rounds will get timeout because when the faulty node is elected it does not send any message.

Result: The nodes keep timing out whenever the fault node is the leader and process with the transaction by obtaining consecutive QCs

Testcase 12 (Full failure: f+1 faulty nodes):

```
{'sq_num':12, 'nclients': 5, 'nreplicas': 3, 'nfaulty': 1, 'timeout': 8, 'numMessages' : 2,
   'failure' : FailureConfig(failures = [
Failure(src=2,dest='',msg_type='',round='_',prob=1,fail_type=FailType.MsgLoss,val=None,
   attr=None)],seed = 12345678), 'delta': 2
   }
```

Description : All faulty nodes do not send any message and nreplica < 3f+1
Outcome: So quorum is never formed in any round and clients keeps timing out and retransmits the request.